A Proactive Transport Protocol for Performance Enhancement of Satellite based Networks

Mohanchur Sarkar
SATCOM Application Area
Space Applications Centre, ISRO
Ahmedabad, India

K.K.Shukla
Department of Computer Engineering
Institute of Technology, BHU
Varanasi, India

K.S.Dasgupta
SATCOM Application Area
Space Applications Centre, ISRO
Ahmedabad, India

ABSTRACT
Current TCP Protocols have lower throughput performance in satellite networks mainly due to the effect of long propagation delays and high link error rates. In this paper a new congestion control protocol for satellite networks is proposed. The protocol uses a proactive approach and is composed of novel ideas like Proactive Slow Start, Proactive Congestion Avoidance and Decision based Error handling policies that are combined with traditional TCP algorithms like Fast Retransmit. The mainstay of our protocol is that the nature of the RTT pattern can give us probable indication of an incipient congestion in the network. This changing pattern of RTT is incidentally used to differentiate between congestion or link error, thus avoiding unnecessary rate throttle. In the initial phase necessary augmentation of ns2 simulator pertaining to the proposed protocol is carried out. This was essential to create a necessary test bed for exhaustive simulation of the protocol considering a GEO network with different congestion level and packet error rate. Simulation results show that the protocol always outperforms other TCP schemes in terms of *goodput* and in cases an improvement of 80% to 120% is observed especially when the packet error rate is very high. Evaluation of the protocol shows a high fairness property and excellent adaptability to high levels of congestion and errors.

General Terms
Algorithms, Performance, Design, Experimentation, Verification.

Keywords
Proactive Protocols, SACK, Vegas, GEO, PER, RTT

1. INTRODUCTION
Over the years, TCP has become the de-facto protocol standard for congestion control in the existing terrestrial Internet. However, experimental and analytical studies [18] confirm that the current TCP protocols variants have performance problems in networks with long propagation delays and relatively high link error rates such as satellite networks [21], [20], [22]. From the view of TCP, the throughput is reciprocal to the *round-trip time* (RTT) of a connection, and is approximately proportional to the congestion window (cwnd), which represents the amount of unacknowledged data the sender can have in transit to the receiver [23]. In satellite networks, TCP throughput decreases because [21], [20], [22] the long propagation delays cause longer duration of the *Slow Start* phase during which the sender may not use the available bandwidth.

One of the major problem in a Satellite based Network is the random packet errors which are not common in the wired counterpart. TCP protocols react to the lack of arrival of acknowledgements or duplicate ACK as a sign of congestion. So the congestion window is reduced which leads to unnecessary throughput degradation. It is a challenge for the network researchers and protocol developers to find means to differentiate the cause of the DUP ACK arrival. Generally probing is done in protocols like Peach [1], Peach+ [2], TP-Planet [3] and RCS [4].

The TCP Protocols can be broadly classified into two categories, reactive protocols and proactive protocols. The reactive protocols do not take any action unless and until the problem really happens. The congestion window is allowed to grow as long as the acknowledgements return, signaling allowable capacity in the network till the point when Duplicate ACKs start coming signifying a loss of packet due to congestion or channel error. At that point corrective actions are taken mainly by reducing the congestion window and slow start threshold by different amounts with the intent to allow the network to come out of the congested state. The reactive protocols with a view to maximize the throughput always drives the network to the maximum capacity after which every connections suffer the collateral damage caused by the overestimation of the channel capacity. All the AIMD TCP protocol variants like Tahoe, Reno, New Reno, SACK, FACK, Peach, Peach+, TP-Planet fall in this category. Reactive algorithms tries to solve the problem but don’t consider why the problem happens. On the other hand the proactive protocol tries to anticipate the overestimation of the network capacity and start taking corrective action to avoid the incipient congestive meltdown of the network. TCP Vegas [5] is a proactive protocol, which is also recommended by the CCSDS SCPS-TP [17] for use in satellite-based networks.

In this paper, we introduce a new congestion control scheme for satellite networks, which is an end-to-end solution to improve the throughput performance in satellite networks. The paper is organized as follows. We introduce the new TCP Protocol in Section2. In Section3, we evaluate the performance of the proposed protocol through simulation. Finally, in Section4 we conclude the paper

2. PROACTIVE CONGESTION CONTROL

The main philosophy of this work is that congestion in a network does not grow all of a sudden. The queue in the routers start growing and that leads to more packet delay or an increase in the experienced RTT. The nature of the pattern of the RTT can give us some indication of the incipient congestion in the network. There exists some empirical value or limits on the RTT, which can signal an incipient congestion. If this changing pattern of the RTT is used to take decision in whether a packet loss is because of congestion or link error, then the unnecessary rate throttle need not be done. We have carried out simulation to verify the impact of congestion and channel errors on the RTT experienced by a connection. First the network is kept within is allowable limit and connections are allowed with congestion window that the network can very well handle and the pattern of RTT is observed. It is seen that the mean RTT remains within a limit. Then the network is slowly congested by increasing the congestion window of the prevailing connections, keeping packet error rates to zero. It is observed that in majority of the cases an increase in mean RTT occurs before a congestive loss is encountered. Then the connections are kept within the capacity and packet error rates are introduced. In this case duplicate acks start coming but the mean RTT is found to be within a limit. This confirmed that the pattern of the RTT could be used to take decisions in a proactive way to avoid congestion and handle errors.

The protocol proposed in this paper is a Proactive protocol and it measures the mean RTT for every congestion window. If an increase in the mean RTT is experienced for three successive congestion window then it can be anticipated that the network is getting in to the congested state. Here the decision-making criteria are that how many congestion window should be checked for successive increase to conclude an incipient congestion. If more number of windows is considered for deriving at the conclusion that the network is moving to a congested state, then the decision may be more accurate but it may lead to a point where the network is so much congested that even taking corrective measures does not help. So it is decided that we take decision based on the mean RTT increase for three successive congestion windows. Out of the three windows the second window will detect an increase and gives us a direction and the next increase will confirm that detection so that corrective action can be taken. If we take a decision just by considering two windows then that may lead to false decisions. We have considered to take the increase in mean RTT rather than considering individual RTT because of the fact that the variation in individual RTT may be attributed by other factors like the choice of the different optimized path by the router, processing delays which are very much temporary and taking decision on that basis creates more frequent oscillations on the congestion window. But when a successive increase in the mean RTT is observed it signals of some major change happening in the network condition.

2.1 Determination of Penalty Factor

After an incipient congestion is detected, the congestion window is reduced by a penalty factor. Now the choice of the penalty factor is very important for the throughput of the protocol, which means how much the window should be decreased so that the incipient congestion can be avoided and the network returns to a stable condition from the congested state. There is a tradeoff, if the penalty is too high then the throughput of the protocol will decrease and if it is too low, then the corrective action necessary for avoiding the congestion will not happen. A simulation has been carried out with different value of the penalty factor from 0.1 to 1 as shown in Fig 1. Penalty factor of 0.1 means a 90% decrease of the congestion window and 1 signified no change to the window, which means not using this logic of proactive congestion avoidance algorithm. The simulation of throughput with different values of penalty factor is plotted and it is seen that with penalty factor 1 the throughput degrades drastically which shows the justification of the proactive approach. A peak in the throughput is at .85 and .90, which corresponds to 15% to 10% decrease. Therefore, the protocol uses a penalty factor of 0.85 with detection of an incipient congestion thereby removing the extra amount of data getting into the network. This will try to keep the overall network load within the tolerable limit and avoid a congestive loss and an eventual congestive meltdown of the network.

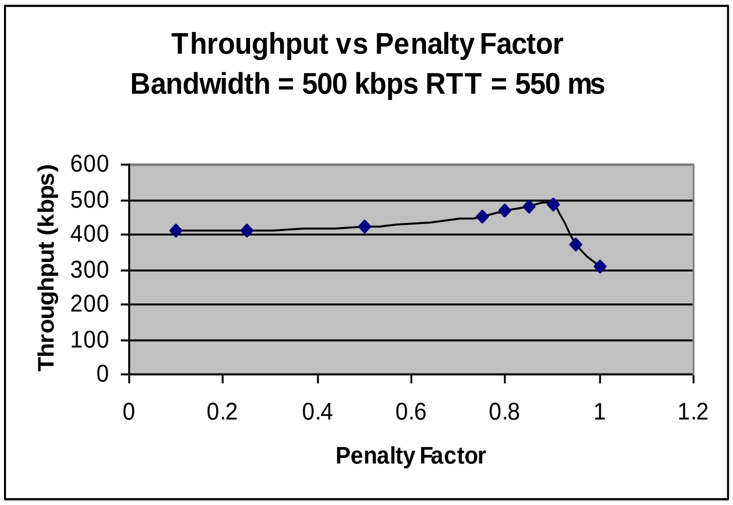

Fig. 1: Evaluation of Throughput for different Penalty Factor

2.2 Proactive Slow Start

In the beginning of a new connection, the sender executes the Slow Start algorithm to probe the availability of bandwidth along the path [27]. During the slow start phase the window doubles itself every RTT and quickly captures the channel bandwidth till the slow start threshold is reached. After that it increases linearly. The slow start phase creates performance degradation in cases the RTT is high. Peach [1], Peach+ [2] uses dummy packets to acquire the available capacity within two RTT by a technique named emulated slowstart. This is an efficient technique but requires special capability of the routers to discard low priority packets. Moreover during time of congestion this will add to the wastage of bandwidth. Vegas [5] uses a more conservative approach by increasing the congestion window every alternate RTT to avoid overestimation of network capacity. In the proposed protocol as the principle is a slow and steady rise of the congestion window we propose to retain the old slow start mechanism. The difference is that Proactive Slow Start phase will consider the incipient congestion algorithm. Before doubling its congestion window it will check if an incipient congestion is detected by an increase in the RTT. In that case the congestion window will not be doubled but rather decreased by the penalty factor.

This will lead to a very controlled way of capturing the network capacity. When a connection starts if the network is not congested slow start will perform normally but if congestion exists in the network, the new connection will not pump too much packet in the network. The logic behind the proactive approach is that the collateral damage has to be avoided i.e. if the network is already getting into congested state a new connection in the process of getting too much resource should not create performance degradation to other connections in the network.

2.3 Decision Based Error Recovery

TCP was initially developed for wire line networks where the link error rate is low, such that the majority of the segment losses are due to network congestions. Thus, the sender assumes that all segment losses are caused by congestions and accordingly it decreases its transmission rate.

Although the application of forward error correction (FEC) algorithms can increase the reliability of satellite links, satellite networks have several orders of magnitude higher error rates than the wireline networks [21]. As a result, we cannot ignore the errors in satellite links and assume that all segment losses occur due to congestion. This assumption may lead to drastic and unnecessary decrease in resource utilization [25], [26], [24], [21].

This problem could be solved if TCP could distinguish whether segment losses occur due to network congestion or due to link errors [20]. However, this is currently infeasible in [21], the authors suggest to decouple error and congestion control. TCP would then be responsible only for congestion control while the error control is handled by the link layer. However, this solution is impractical because the link layers of all sub networks composing the Internet need to be redesigned. An alternative solution is that the sender could contain an algorithm, which can distinguish between congestion and errors.

However, such an algorithm must be very reliable. In fact, if this algorithm does not respond correctly to actual network congestion, the network utilization decreases drastically [21]. To our knowledge, such a reliable algorithm does not exist to date.

Peach [1], Peach+ [2], TP-Planet [3] distinguishes errors and congestion by physically probing the network to confirm whether there is capacity in the network by sending low priority dummy packets and monitoring the reception of their Acknowledgement. These dummy packets add a high overhead of 17% [1] on the satellite channel and require special capability of the routers to drop low priority packets. This leads to wastage of the precious satellite bandwidth and its deployment also requires a change in routers along the path.

In the proposed protocol we have considered the protocol to be moving through two broad states. An incipient congestion state which is signaled by the increase in RTT and an un-congested state. This state information is used by the protocol to handle the losses due to channel errors. So whenever three duplicate ACKs are received the protocol will check whether the protocol is in an incipient congestion state or not. If yes the congestion window and slow start threshold will be reduced to half. But if there is no sign of an incipient congestion the congestion window and slow start threshold will be kept unchanged only a Fast Retransmit will be done for the packet lost. Then the congestion window can increase linearly with each received ACK.

It has been seen that a considerable amount of time is taken to gain an appreciable congestion window. So in this protocol the approach we have taken is that we will increase very conservatively and once an appreciable value is reached the protocol will proactively try to sense the network condition by measuring the mean RTT values and take action at the earliest opportunity. It will also try to avoid a reduction in congestion window if the loss is caused by channel error so that the protocol can maintain the gained congestion window value.

2.4 Overall Working of Proactive TCP

In Fig 2 the overall mechanism of the proposed Proactive TCP scheme is depicted. The protocol starts with the Proactive Slow Start phase where the slow start threshold is kept at half the receiver window, the congestion window to one and the state of the protocol is un-congested state. The receiver window signifies the maximum number of segments the receiver can accommodate. The congestion window is increased by one with each received ACK so that the congestion window doubles every RTT until the slow start threshold is reached. However, the mean RTT is also checked during this phase and if RTT increase is detected the congestion window is reduced by the penalty factor. After slow start threshold is crossed the protocol moves to the Proactive Congestion Avoidance phase where it increases the congestion window by 1/cwnd for every reception of ACK as in traditional Congestion Avoidance Algorithm.

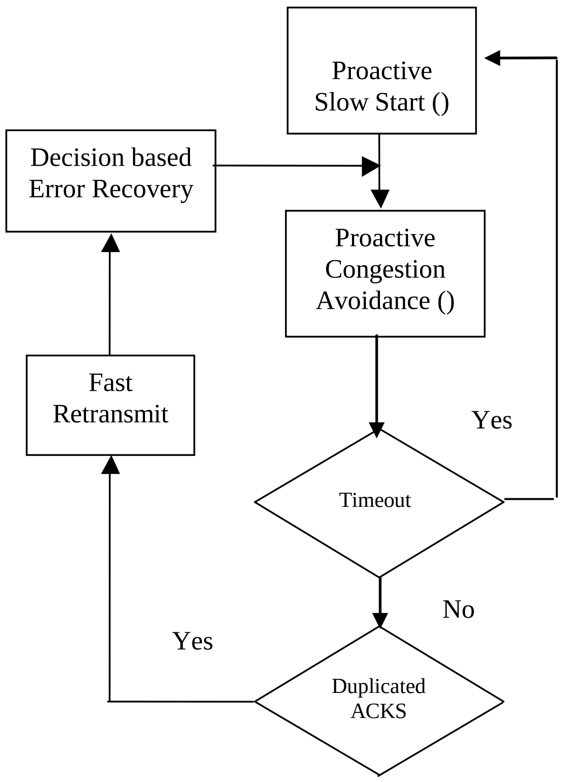

Fig. 2: Proactive TCP Scheme

If an acknowledgement is not received for a transmitted segment within its retransmission timeout period, the timer expires and the Proactive Slow Start phase is again initiated. During the Proactive Congestion Avoidance phase, the mean RTT is checked every time and if an increase in RTT for three successive congestion windows is detected; the congestion window is reduced by the penalty factor, and the state of the protocol changes to incipient congestion state. After this if three duplicate ACKs are received the protocol calls the Fast Retransmit algorithm, a retransmission of the lost segment is done, and the protocol moves to the Decision based Error Recovery Phase. In this phase if the protocol is in an incipient congestion phase, which signals that the network is in a congested state and a packet is not received by the receiver then the chance of the packet being lost by congestion is more.

Moreover it is seen that even though the Proactive Congestion Avoidance algorithm has already taken action to avoid congestion in the network the segment loss is detected. So the congestion window is reduced to half and protocol moves to the Proactive Congestion avoidance phase. If the state of the protocol is in un-congested state then no reduction in congestion window is done and protocol moves to Proactive Congestion avoidance as it is anticipated to be a loss due to error in the channel. The state of a protocol is changed from incipient congestion state to un-congested state when a decrease in RTT for three successive congestion windows is observed. In case that a segment is retransmitted by Fast Retransmit, but again it gets lost or its acknowledgement gets lost because of either congestion or corruption the retransmission timer will expire which signals a major problem in the network and the congestion window is reduced to one and slow start threshold reduced to half of its previous value and Proactive Slow Start starts which will increase the congestion window in a fast but conservative manner to adapt to the available capacity.

The algorithms described above are implemented in ns2 Simulator [16] inheriting the in built objects available in ns2 and adding the necessary functionality.

3. SIMULATION AND ANALYSIS

We evaluate the performance of the proposed TCP in terms of goodput and fairness through simulations when several connections share the same link. We simulate the system as shown in the Fig3 below where N senders transmit data to N receivers through a satellite channel. The N streams are multiplexed in Earth Station A, whose buffer can accommodate K segments. The segments may get lost with a packet error rate PER. In this experiment all the N senders are each connected to the Earth station A with a link of bandwidth 500kbps and RTT of 10ms. All the N receivers are connected to Earth station B with a 500kbps link with RTT 10ms. We have taken N = 10, K = 25 segments, receiver window rwnd = 64 segments, the link between Earth Station A to B via satellite to be 5Mb and the RTT between the two stations as 550 ms.

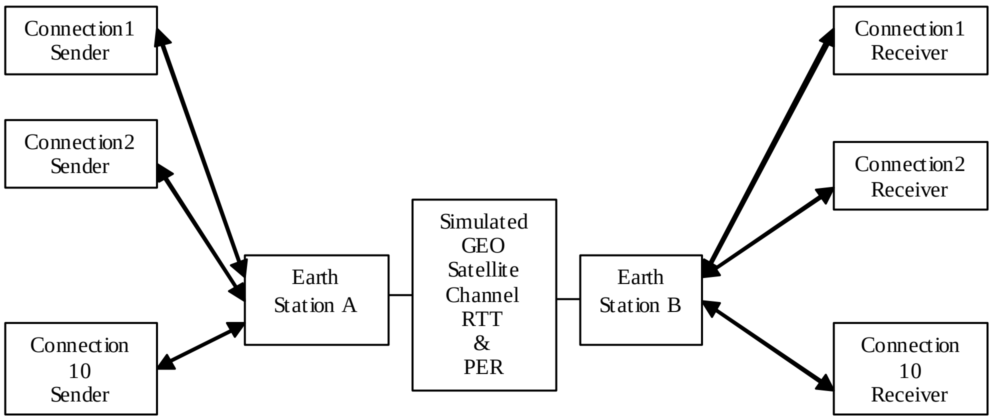

Fig. 3: Simulation Scenario

All the results in this section have been obtained by considering the system behavior for 550s, which is 1000 times the round trip time value. All the ten connections share a common satellite channel between Station A and B with a link capacity of five Mbps. In the Fig4, the throughput obtained by the individual connections for increasing RTT is plotted. The plot corresponds also to different PER values ranging from 0.001 to 0.1.It can be seen that the throughput degrades with increasing values of RTT. It can also be seen that when the RTT is small even a high PER does not significantly degrade the throughput. In the left hand it can be seen that the curves for different PER are closer on the other hand with high RTT the curves all highly dependent on the PER rates also. Another significant advantage of the protocol is that even with high error rates of 10^{-2} and 10^{-1} the throughput is not degraded drastically.

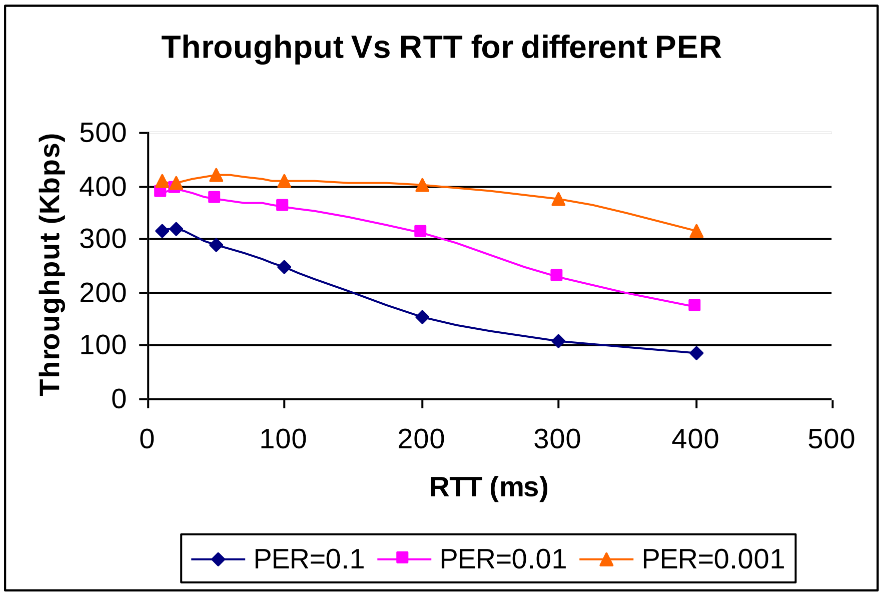

Fig. 4: New TCP Throughput for varying RTT for different PER

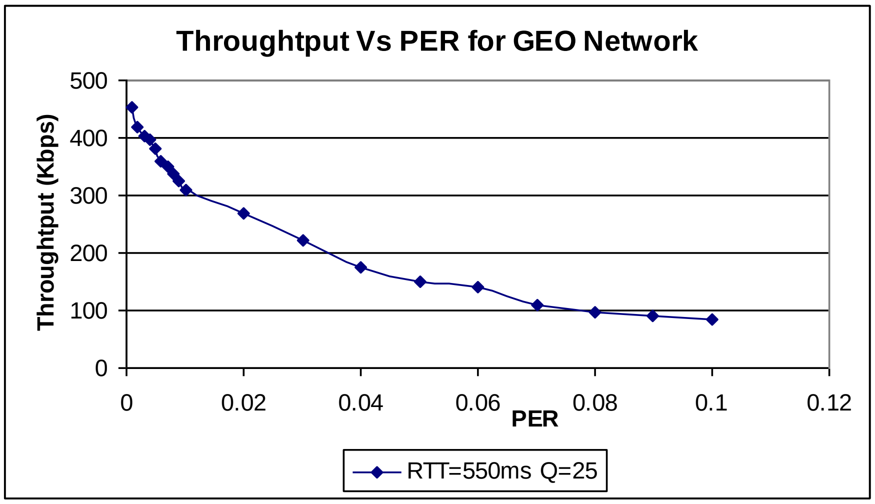

Fig. 5: New TCP Throughput for increasing PER

In the test simulation 10 senders communicating to 10 receivers each using 500kbps channel and eventually multiplexed in an Earth station and sent to a GEO satellite with bandwidth of 5Mbps is used for the generation of the plot shown in above Fig 5. The Packet Error Rate of the link between Earth Station A to B is varied from 0.001 to 0.1. The mean throughput obtained by all the 10 connections is used in generating the graph. From the graph it can be seen that the throughput is quite high is case of PER less than 0.01. Theoretically each connection can get a maximum throughput of 500kbps as 10 connections share a 5Mb bottleneck link and from the graph it can be seen that more than 300 kbps is achieved for PER of less than 0.01. The high significant reduction in throughput occurs when the PER approaches 0.1 is very uncommon as the BER of the channel is generally maintained with FEC to have PER of 0.01.

3.1 Goodput Evaluation with TCP SACK and TCP Vegas

Goodput is the effective amount of data delivered through the network. It is a direct indicator of network performance. We expect that a good TCP scheme transmit as much data as possible, while behaving friendly to other TCP flows in terms of consuming the network resource, e.g., bandwidth. In the following graphs shown the throughput of the protocol is compared to TCP SACK [6] and TCP Vegas [5] in the same test bed of 10 senders communicating to 10 receivers using a 5Mb bottleneck link via satellite. To artificially induce congestion the bandwidth between Earth Station A and B is stepwise reduced to see the reaction of the protocol to congestion. Here all ten different connections are connected to the Earth Station with 500 kbps bandwidth.

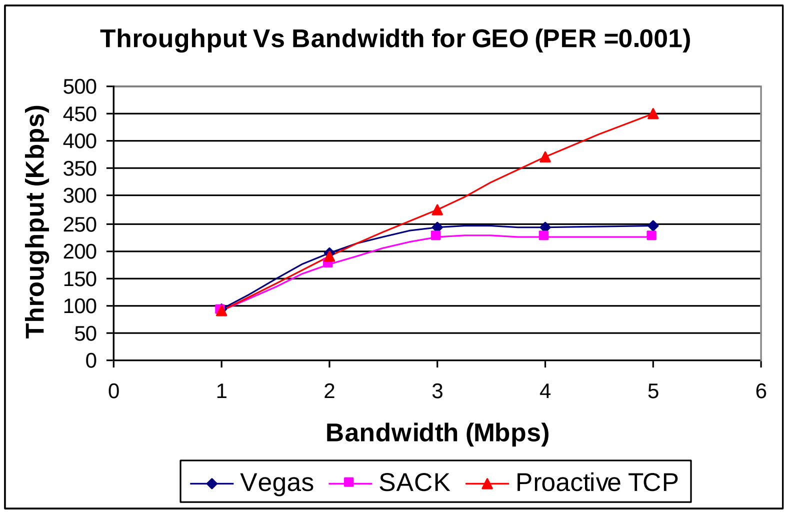

Fig. 6: Throughput with Bandwidth PER = 0.001

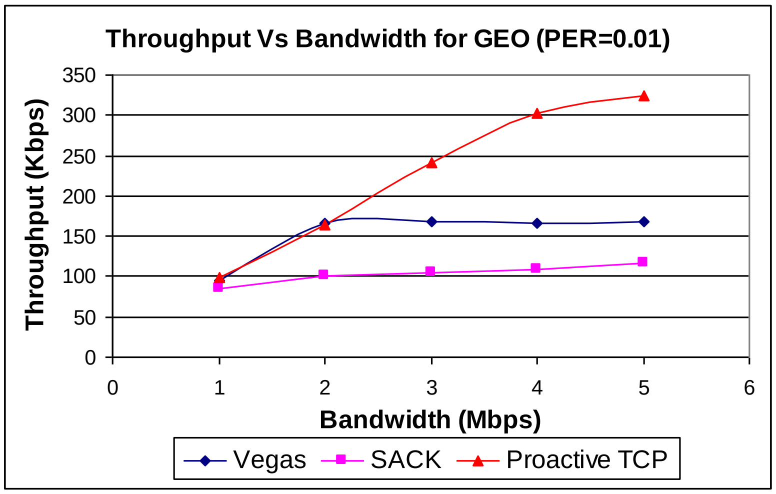

Fig. 7: Throughput with Bandwidth PER = 0.01

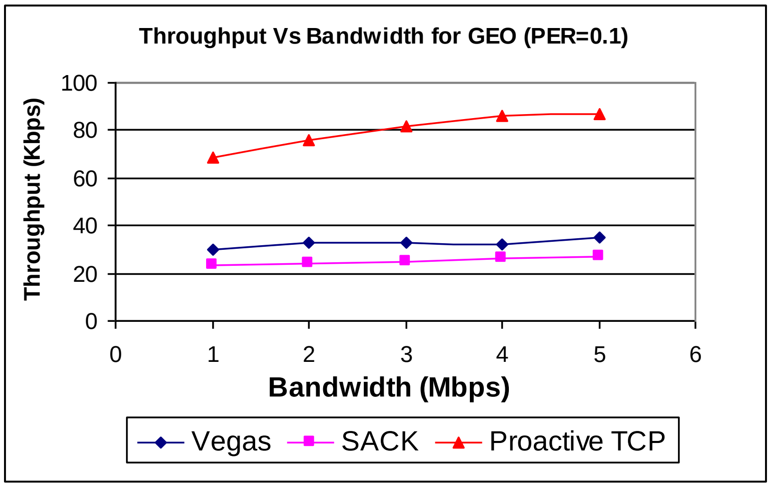

Fig. 8: Throughput with Bandwidth PER = 0.1

Therefore, the minimum aggregate bandwidth required for all the connections to perform optimally is 5Mbps. When we reduce the satellite link bandwidth this will lead to congestion in the network and it is the job of the protocol to handle the congestion and reduce the data rate and adapt to the changing network condition and available bandwidth. In the first case, a PER of 0.001 is assumed and it can be seen from Fig 6 that at 5Mbps bandwidth, which signifies the minimum capacity, needed, the proposed TCP outperforms TCP SACK and TCP Vegas. In all the figures it can be noticed than when the congestion is very high as 1Mbps bottleneck link the protocol performs smoothly and gets a throughput close to 100 kbps so the effect of congestion is removed. In Fig 6, it can be seen as we go on reducing the bandwidth the protocol adapts to the new available capacity outperforming the SACK and Vegas. This is possible because of the proactive nature of the protocol, which detects an incipient congestion very quickly and takes corrective measures. When PER of 0.1 is used as shown in Fig. 8 it can be seen that the protocol outperforms SACK and Vegas by 100%. This is because of the fact that the protocol uses the prediction logic for differentiating the cause of loss of packet or arrival of duplicate ACKs between congestion and error.

3.2 Impact on the congestion window

Normally in TCP Protocols, the congestion window is increased with each successful delivery of data, which leads to a larger share of the bandwidth available for the corresponding connection, until a point is reached that the network gets so overloaded than its capacity that packets are dropped from the router queue and the congestion window is drastically reduced. It may also happen that the network is not congested, but due to channel error, data is lost and TCP unable to detect the cause of the error reduces the congestion window. Therefore, the congestion window grows initially remains at a certain level for some time and then falls drastically.

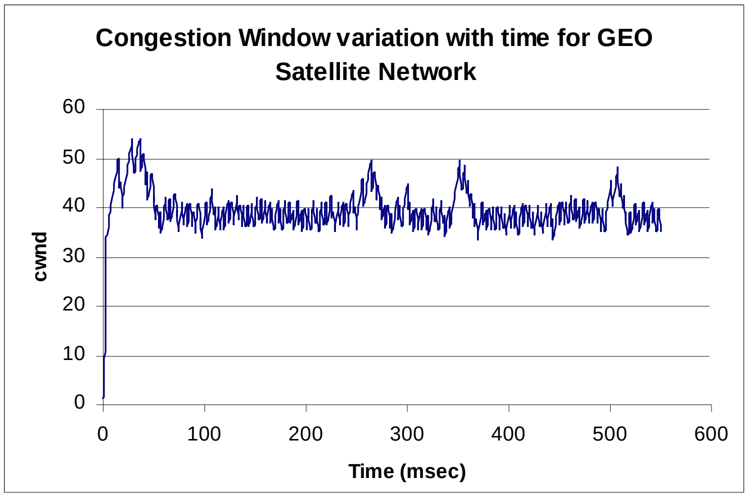

Fig. 9: Congestion Window Variation for GEO

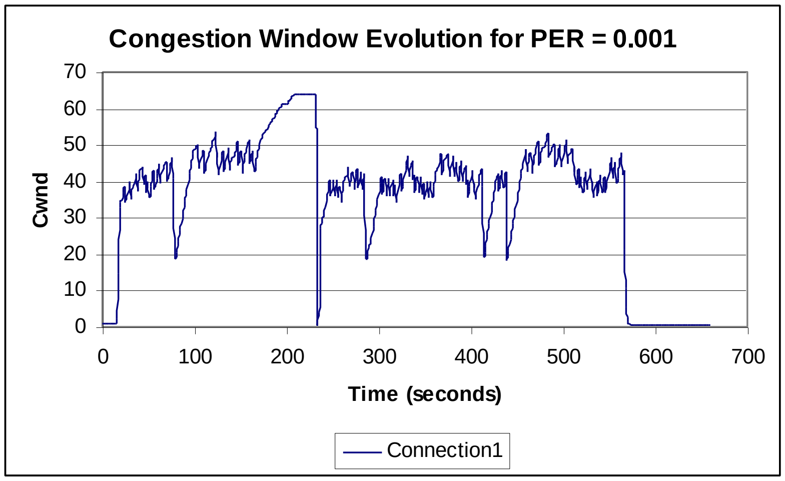

Fig. 10: Congestion Window for GEO for PER = 0.001

This total time called the congestion epoch determines the throughput of the connection. The larger the congestion epoch better is the utilization of the channel capacity. In Fig 9 is shown the behavior of the congestion window for the proposed protocol and it is seen that the congestion epoch is increased appreciably. Due to the proactive nature of the protocol, the congestion window during an excessive congested condition is found to oscillate below its maximum allowable limit. This gives the necessary performance enhancement the protocol achieves than its peers. Moreover, in this proposed protocol an attempt has been made to distinguish between congestive and corruptive loss. Therefore, in cases where a packet is lost due to channel error the protocol detects the channel error and no reduction is done on the congestion window. This retains the value of the congestion window and avoids unnecessary rate throttle.

In Fig 10, it can be seen that the congestion window drops at certain points, which is attributed when duplicate ACKS are received when the protocol is in an incipient congestion phase. In the protocol the timeout algorithm is not changed, the binary backoff algorithm is used which doubles the timeout value with every timeout expiry. The timeout algorithm is kept unchanged keeping in view of its stable and proven nature and in case, that there in a false alarm in the decision making process the congestion window will be reduced with the timeout expiry. This will keep the network capacity under control in cases of heavy congestion or high error condition. The timer expiry will overrule the decisions taken by the proactive algorithm. But it can be seen that the congestion epoch is appreciable increased in Fig 10 which is attributed by the decision based error recovery and proactive congestion control algorithms.

3.3 Fairness of Proposed TCP

The fairness of a protocol is ability of the connection to equally share the available bandwidth so that it does not happen that the connection, which starts first, takes an appreciable proportion of the bandwidth and does not release the capacity even when other connections are active in the network. This property is very much desirable in any new protocol and in the process of enhancement of throughput the fairness property should not be violated. So multiple connections of the same TCP scheme must interoperate nicely and converge to their fair share. The fairness index function (1) is proposed in [15], to justify the fairness of TCP schemes. The fairness index function is expressed as

$$F(x) = \frac{(\sum_{i=1}^{N} x_i)^2}{N * (\sum_{i=1}^{N} x^2{}_i)} \qquad (1)$$

Where x_i is the throughput of the ith connection, and n is the number of connections. F(x) ranges from 1/n to 1.0. A perfectly fair bandwidth allocation would result in a fairness index of 1.0. On the contrary if all bandwidth were consumed by one connection, (1) would yield 1/n. In Table 1 the fairness index calculated from all the 10 connections for different link capacity and packet error rate is shown. It can be seen that all the connections get an almost equal chare of bandwidth [14]. In Table 1, the columns correspond to decreasing levels of bandwidth of the satellite link, which will create increasing congestion in the routers as the data received in an Earth Station is not getting the needed bandwidth to deliver the data to the destination. The rows correspond to increasing rates of packet error starting from no error to a PER of 0.1 where 1 out of 10 packets get corrupted. The protocol is seen to maintain a high degree of fairness even during high degree of congestion and error only a very slight degradation of fairness with increasing congestion or packet error can be seen.

Table1 Jain's Fairness Index for different bandwidth and packet error rate

	5 Mb	4Mb	3Mb	2 Mb	1 Mb
000	.99972	.99883	.99661	.99196	.98833
.001	.99918	.99810	.99628	.96653	.97068
.01	.99372	.99361	.99639	.99504	.98621
.1	.99999	.99871	.99289	.99495	.98834

4. CONCLUSION

The distribution of RTT in normal network condition and a congested network condition is evaluated. As discussed in this paper we propose a new TCP scheme to improve the performance of TCP in Satellite based networks. The proactive TCP Protocol proposed is found to outperform its peers like Vegas and SACK under different levels of congestion and channel errors. The proactive nature of the protocol makes it quite robust in handling high degree of congestion where the protocol is found to well adapt to the available capacity. The advantage obtained by using Proactive Slow Start is that connections initiated during congested network conditions are restricted of creating collateral damage to other ongoing connections. The Decision based error recovery is very effective in not letting the congestion window degrade by differentiating errors and congestion making the protocol suitable for use in satellite based networks. Simulation results show that the protocol always outperforms other TCP variants, in terms of goodput by 80% to 120% when the packet error rate is high. The most important merit is that the performance improvement obtained in this protocol [19] does not need any change in the Routers and all changes are restricted to the sender and receiver protocol stack. Moreover there is no extra overhead associated with the protocol for taking the decisions. The protocol is found to have the very good fairness property as analyzed in the paper. Though we have evaluated the protocol for GEO satellite network the protocol will also perform well in LEO or MEO based satellite networks. It can also be used in the terrestrial Internet for its good proactive congestion control and slow start mechanism.

5. ACKNOWLEDGMENT

Authors thank all members of Department of Computer Engineering, I.T. BHU, and all staff members of Advanced Communication Technology Division (ACTD), SAC/ISRO for their cooperation and technical support.

6. REFERENCES

[1] Ian F. Akyildiz, Giacomo Morabito, Sergio Palazzo, IEEE ACM T NETWORK. 2001. TCP-Peach: A New Congestion Control Scheme for Satellite IP Networks IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 9, NO. 3, JUNE 2001

[2] Ian F. Akyildiz, Xin Zhang, Jian Fang IEEE COMMUN LETT. 2002. TCP-Peach+: Enhancement of TCP-Peach for Satellite IP Networks, IEEE COMMUNICATIONS LETTERS, VOL. 6, NO. 7, JULY 2002

[3] Ozgur B. Akan, Jian Fang, Ian F. Akyildiz, IEEE J SEL AREA COMM. 2004. TP-Planet: A Reliable Transport Protocol for InterPlaNetary Internet IEEE/SAC, vol 22, no 2, Feb. 2004, pp 348-61

[4] Jian Fang and Özgür B. Akan, IEEE COMMUN LETT. 2004. Performance of Multimedia Rate Control Protocols in InterPlaNetary Internet, IEEE COMMUNICATIONS LETTERS, VOL. 8, NO. 8, AUGUST 2004

[5] L. S. Brakmo, S. O Malley, L. L. Peterson, COMP COMR 1994. TCP Vegas: New Techniques for Congestion Detection and Avoidance, *Proc. ACM SIGCOMM 1994*, pp. 24-35, October 1994

[6] Mathis, M., J. Mahdavi, S.Floyd, and A.Romanow, TCP Selective Acknowledgement Options. RFC 2018, April 1996

[7] Bonald, T., Comparison of TCP Reno and TCP Vegas: Efficiency and Fairness. In Proceedings of PERFORMANCE 99, Istanbul, Turkey, October 1999

[8] C. Casetti, M. Gerla, S. Mascolo, M.Y. Sanadidi, R. Wang, TCP Westwood: bandwidth estimation for enhanced transport over wireless links, Proceedings of Mobicom 2001, Rome, Italy July (2001)

[9] M. Mathis, J. Mahdavi, S. Floyd, A. Romanov, "TCP Selective Acknowledgement Options," *RFC 2018*, 1996.

[10] M. Mathis, J. Mahdavi, COMP COMR. 1996. Forward Acknowledgement: Refining TCP Congestion Control, *Proc. ACM SIGCOMM 1996*, pp. 281-292, Aug. 1996.

[11] J. Padhye, V. Firoio, D. Towsley, J. Kurose, IEEE ACM T NETWORK 2000. Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation, *IEEE/ACM Trans. Networking*, Vol. 8, No. 2, pp. 133-145, April 2000.

[12] P. Sinha, N. Venkitaraman, R. Sivakumar, V. Bharghavan, *ACM MOBICOM 1999.* WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks. *Proc.ACM MOBICOM 1999*, pp. 231-241, Seattle, Washington, August 1999.

[13] D. T. Tran, F. J. Lawas-Grodek, R. P. Dimond, W. D. Ivancic, SCPSTP, TCP and Rate-Based Protocol Evaluation for High-Delay, Error-Prone Links, *SpaceOps 2002*, Houston, TX, October 2002.

[14] S.C. Tsao, Y.C. Lai, and Y.D. Lin, IEEE NETWORK. 2007. Taxonomy and Evaluation of TCP-Friendly Congestion-Control Schemes on Fairness, Aggressiveness, and Responsiveness, IEEE Network, November 2007

[15] R. Jain, D. Chiu, and W. Hawed, A quantitative measure of fairness and discrimination for resource allocation in shared computer systems, DEC, Res. Rep.TR-301, 1984.

[16] UCB/LBNL/VINT Network Simulator [Online]. Available: http://www.isi.edu/nsnam/ns/

[17] Consultative Committee for Space Data Systems, "Space Communications Protocol Specification-Transport Protocol (SCPS-TP)", *Recommendation for Space Data Systems Standards, CCSDS 714.0-B-1.*, Blue Book. Issue 1, Washington, D.C.: CCSDS, May 1999.

[18] T. V. Lakshman and U. Madhow, IEEE ACM T NETWORK. 1997. The performance of TCP/IP for networks with high bandwidth-delay products and random loss, *IEEE/ACM Trans. Networking*, vol. 5, June 1997.

[19] Injong Rhee and Lisong Xu, IEEE ACM T NETWORK, 2007, Limitations of Equation-based Congestion Control

[20] C. Metz, IEEE INTERNET COMPUT. 1999. TCP over satellite the final frontier *IEEE Internet Comput.*, pp. 76–80, Jan. /Feb. 1999.

[21] C. Partridge and T. J. Shepard, IEEE NETWORK. 1997. TCP/IP performance over satellite links, *IEEE Network Mag.*, pp. 44–49, Sept/Oct. 1997.

[22] M. Allman *et al.*, Ongoing TCP research related to satellites, RFC 2760, Feb. 2000.

[23] W. Stevens, *TCP/IP Illustrated*. Reading, MA: Addison-Wesley, 1994, vol.1

[24] R. Fox, TCP big window, and NAC options, Request for Comments 1106, IETF, June 1989.

[25] H. Bal Krishnan, S. Seshan, E. Amir, and R. H. Katz, Improving TCP/IP performance over wireless networks, *Proc. ACM Mobicom*, Nov. 1995, pp. 2–15.

[26] S. Dawkins, G. Montenegro, M. Kojo, V. Magret, and N. Vaidya, End-to-end performance implications of links with errors, Internet draft, Mar. 2000.